WHAT IS CLAIMED IS:

1. A method for managing memory available for dynamic allocation during execution of code containing a plurality of memory allocators and a plurality of memory deallocators, comprising:

allowing a user to establish a relationship between one or more of the memory deallocators and one or more of the memory allocators, wherein the relationship requires that memory space allocated by the one or more allocators is freed by the one or more deallocators;

allowing the code to execute;

upon a call to the one or more deallocators to free a memory space, determining whether the relationship is violated; and

if so, notifying the user.

2. The method of claim 1, wherein notifying the user comprises halting execution of the code.

3. The method of claim 1, wherein notifying the user comprises halting execution of the code and displaying a status message to the user.

4. The method of claim 1, if the relationship is not violated, freeing the memory space.

5. The method of claim 1, wherein determining whether the relationship is violated comprises determining that the memory space was allocated by an allocator different from the one or more memory allocators.

6. A method for managing memory available for dynamic allocation during execution of code containing a plurality of memory allocators and a plurality of memory deallocators, comprising:

establishing a relationship between a user-selected memory deallocator and a user-selected memory allocator, wherein the relationship requires that memory space freed by the user-selected deallocator have been allocated by the user-selected allocator;

allowing the code to execute;

upon a call to the user-selected deallocator to free a memory space, determining whether the memory space was allocated by the user-selected allocator; and

if so, notifying the user that the relationship is violated.

7. The method of claim 6, wherein notifying the user comprises halting execution of the code and displaying a status message to the user.

8. A method for managing memory available for dynamic allocation during execution of code containing a plurality of memory allocators and a plurality of memory deallocators, comprising:

setting an upper limit on the amount of memory space an allocator can allocate during execution of the code, wherein the upper limit is specific to the allocator;

during execution of the code, tracking the amount of memory space allocated by the allocator; and

when the amount of memory space allocated exceeds the limit, notifying a user.

9. The method of claim 8, wherein the step of tracking comprises:

determining whether the allocator is called to allocate memory and, if so, incrementing a counter; and

determining whether a deallocator is called to deallocate memory allocated by the allocator and, if so, decrementing the counter.

10. The method of claim 8, wherein the step of tracking comprises incrementing a counter in the event of memory allocation by the allocator and decrementing the counter in the event of memory deallocation of memory space allocated by the allocator.

11. The method of claim 8, wherein notifying the user comprises halting execution of the code.

12. The method of claim 8, wherein the upper limit is independent of other memory size limitations.

13. The method of claim 8, wherein the upper limit is not a limit on a stack size.

14. A computer readable medium containing a program which, when executed, performs an operation for managing memory available for dynamic allocation during execution of code containing a plurality of memory allocators and a plurality of memory deallocators, the operation comprising:

establishing a relationship between a user-selected memory deallocator and a user-selected memory allocator, wherein the relationship requires that memory space freed by the user-selected deallocator have been allocated by the user-selected allocator;

allowing the code to execute;

upon a call to the user-selected deallocator to free a memory space, determining whether the memory space was allocated by the user-selected allocator; and

if so, notifying the user that the relationship is violated.

15. The computer readable medium of claim 14, wherein notifying the user comprises halting execution of the code and displaying a status message to the user.

16. A computer readable medium containing a program which, when executed, performs an operation for managing memory available for dynamic allocation during execution of code containing a plurality of memory allocators and a plurality of memory deallocators, the operation comprising:

setting an upper limit on the amount of memory space an allocator can allocate during execution of the code, wherein the upper limit is specific to the allocator;

during execution of the code, tracking the amount of memory space allocated by the allocator; and when the amount of memory space allocated exceeds the limit, notifying a user.

17. The computer readable medium of claim 16, wherein the step of tracking comprises:

determining whether the allocator is called to allocate memory and, if so, incrementing a counter; and

determining whether a deallocator is called to deallocate memory allocated by the allocator and, if so, decrementing the counter.

18. The computer readable medium of claim 16, wherein the step of tracking comprises incrementing a counter in the event of memory allocation by the allocator and decrementing the counter in the event of memory deallocation of memory space allocated by the allocator.

19. The computer readable medium of claim 16, wherein notifying the user comprises halting execution of the code.

20. The computer readable medium of claim 16, wherein the upper limit is independent of other memory size limitations.

21. The computer readable medium of claim 16, wherein the upper limit is not a limit on a stack size.

22. A computer system comprising an output device, a memory device, one or more processors, code resident in the memory device and containing a plurality of memory allocator calls and a plurality of memory deallocator calls, a heap manager resident in the memory device to allocate and free memory of the memory device and a debugger program resident in the memory device; the debugger program comprising a debugger user interface configured to at least:

allow a user to view allocation/deallocation history information at a user-specified memory location; and allow a user to establish a relationship between a memory deallocator call and a memory allocator call, wherein the relationship requires that memory space allocated by the memory allocator call is freed by the memory deallocator call and a violation of the requirement causes the debugger user interface to notify the user.